(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2002－204356
（P2002－204356A）
(43)公開日　平成14年７月19日(2002.7.19)

(51)Int.Cl.[7]	識別記号	FI		テーマコード(参考)
H04N 1/41		H04N 1/41	B	5C059
H03M 7/40		H03M 7/40		5C078
H04N 7/30		H04N 7/133	Z	5J064

審査請求　未請求　請求項の数９　ＯＬ　（全 12 頁）

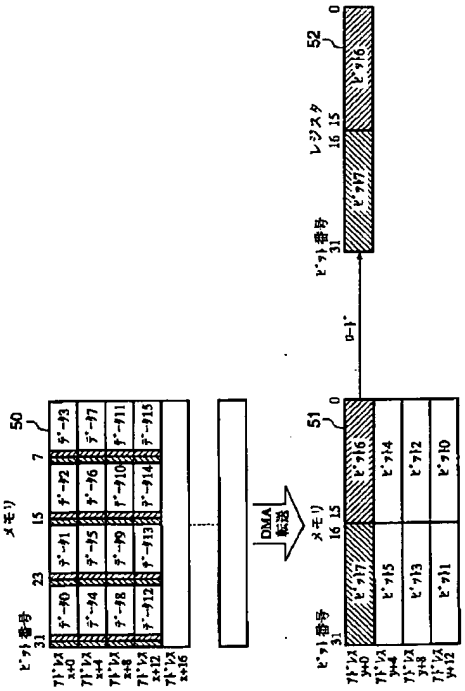
(21)出願番号　特願2001－297445(P2001－297445)
(22)出願日　平成13年９月27日(2001.9.27)
(31)優先権主張番号　特願2000－329421(P2000－329421)
(32)優先日　平成12年10月27日(2000.10.27)
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号
(72)発明者　大佐　欣也
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内
(74)代理人　100076428
弁理士　大塚　康徳　（外３名）

Ｆターム(参考)　5C059 MA00 MA24 MA35 MC11 ME11 UA15 UA30 UA36
5C078 BA57 BA58 CA31 DA01 DA02 DB19
5J064 AA02 AA03 BA09 BA16 BC01 BC11 BD01

(54)【発明の名称】　データ処理装置、プロセッサ、及びその制御方法

(57)【要約】
【課題】　ビットデータ、特に多値データのビットプレーンの転送の高速化、効率化を行うこと。
【解決手段】　メモリ５０には、８ビットの多値データが１ワードにつき４個ずつ格納されており、４×４＝１６個の多値データでビットプレーン符号化のひとつの処理単位（処理ブロック）をなしているとする。メモリ領域５１では、各多値データ（図５では、データ０乃至データ１５）の最上位ビット（ビット７）が各多値データ順に集められて、斜線部で示されるように１箇所にまとめて格納される。ビット６についても、同様に格納される。



1

【特許請求の範囲】

【請求項１】　所定のメモリ間でデータ転送を行うデータ処理装置であって、
複数のビットにより構成される多値データが第１のメモリに複数存在し、当該複数の多値データから同一ビットプレーンに属するビットからなるビットプレーンデータ群を、当該ビットプレーンデータ群単位で第２のメモリにデータ転送することを特徴とするデータ処理装置。

【請求項２】　所定のメモリ間でデータ転送を行うデータ処理装置であって、
複数のビットプレーンデータ群が存在し、各ビットプレーンデータ群において同一の多値データに属するビットデータを所定の順番で選択することで、多値データを生成することを特徴とするデータ処理装置。

【請求項３】　ビットプレーン符号化処理を含む符号化処理を行う符号化装置において、請求項１に記載のデータ処理装置が含まれていることを特徴とする請求項１又は２に記載のデータ処理装置。

【請求項４】　前記符号化処理はＪＰＥＧ２０００であることを特徴とする請求項３に記載のデータ処理装置。

【請求項５】　前記データ処理装置は、ＤＭＡ回路であることを特徴とする請求項１乃至４のいずれか１項に記載のデータ処理装置。

【請求項６】　所定のメモリから所定のレジスタにデータをロードするプロセッサであって、
複数のビットにより構成される多値データが前記所定のメモリに複数存在し、当該複数の多値データから同一ビットプレーンに属するビットからなるビットプレーンデータ群を当該ビットプレーンデータ群単位で前記所定のレジスタにロードすることを特徴とするプロセッサ。

【請求項７】　所定のメモリ間でデータ転送を行うデータ処理装置の制御方法であって、
複数のビットにより構成される多値データが第１のメモリに複数存在し、当該複数の多値データから同一ビットプレーンに属するビットからなるビットプレーンデータ群を、当該ビットプレーンデータ群単位で第２のメモリにデータ転送することを特徴とするデータ処理装置の制御方法。

【請求項８】　所定のメモリ間でデータ転送を行うデータ処理装置の制御方法であって、
複数のビットプレーンデータ群が存在し、各ビットプレーンデータ群において同一の多値データに属するビットデータを所定の順番で選択することで、多値データを生成することを特徴とするデータ処理装置の制御方法。

【請求項９】　所定のメモリから所定のレジスタにデータをロードするプロセッサの制御方法であって、
複数のビットにより構成される多値データが前記所定のメモリに複数存在し、当該複数の多値データから同一ビットプレーンに属するビットからなるビットプレーンデータ群を当該ビットプレーンデータ群単位で前記所定の

2

レジスタにロードすることを特徴とするプロセッサの制御方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、所定のメモリ間でデータ転送を行うデータ処理装置、所定のメモリから所定のレジスタにデータをロードするプロセッサ、及びその制御方法に関するものである。

【０００２】

【従来の技術】特開平０８－１４９３０８号公報や特開平０９－０２７７５２号公報に示されるように、画像符号化の方式としてＷａｖｅｌｅｔ変換後の係数値を、ビットプレーン符号化（ビットプレーン単位でのエントロピー符号化）する方式が提案されており、このような方式は、ＩＳＯにて標準化の進められているＪＰＥＧ２０００符号化方式において、採用が予定されている。

【０００３】この画像符号化方式の処理の流れを一般化して示すと図２のようになる。図２おいて入力画素値２０は、ひとつひとつの画素の値が複数のビットで表される多値データあり、これをＷａｖｅｌｅｔ変換（あるいはＤＣＴなどの他の変換符号化方式）にて変換符号化（２１）処理すると、その出力は係数値（２２）となる。係数値２２もやはり多値データである。同図では省略してあるが、必要に応じてこの係数値２２の量子化を行ってもよい。次にこの係数値２２（もしくは、この係数値を量子化した結果の量子化値）をビットプレーン単位でエントロピー符号化（２３）し、符号化データ（２４）を得る。

【０００４】ビットプレーン符号化においては、入力となる多値データを、ビットプレーンを単位として処理しなければならないため、入力多値データ配列を個々のビットプレーンに分解する処理が論理的には必要となる。図３は、４×４の４ビット係数値の配列３０を４つのビットプレーン３１，３２，３３，３４に分解する処理の例を示している。

【０００５】ここで、メモリ上に格納された多値データを、符号化プロセッサによって読み込んで（レジスタにロードして）ビットプレーン符号化を行う実装を考える。

【０００６】メモリのデータバス幅をｗビット、多値データのビット幅（深さ）をｄビットとし、メモリ上にはｄビットの多値データが隙間なく順に詰まっているとする。

【０００７】符号化プロセッサがこのメモリに対して直接ロードを行ったとき、１回のロードで読み込めるデータｗビットの中には、ｗ／ｄ個分の多値データが含まれているが、ひとつのビットプレーンだけに着目すると、そのビットプレーンに含まれるビットは、ｗ／ｄビットだけであり、このビットプレーンを以下の処理に用いる対象とする場合、残りの（ｗ－ｗ／ｄ）ビットは無駄な

ビットとなる。従って、１枚のビットプレーンに対するビットプレーン符号化処理を行うためには、何度もビットプレーンのロードを繰り返し、不要なビットをマスクして必要なビットを取り出す必要がある。

【０００８】図４に、前述の符号化処理において、符号化プロセッサのレジスタ幅（１ワード）＝３２ビット、メモリデータバス幅Ｗ＝３２ビット、多値データのビット幅ｄ＝８ビットとした場合の例を示す。メモリ４０には、８ビットの多値データが１ワードにつき４個ずつ格納されており、４×４＝１６個の多値データでビットプレーン符号化のひとつの処理単位（処理ブロック）をなしているとする。多値データの最上位ビット（図４の斜線部）により構成されるビットプレーンをビットプレーン符号化処理の対象とする場合を考えると、まず、データ０乃至３の最上位ビット３１、２３，１５，７の４ビットをレジスタ４１にロードするために、メモリ４０のアドレス０から１ワード分のロード処理を行う。

【０００９】ここで、ロードしたデータ０乃至３のうち、ビットプレーン符号化に必要なビットは、上記４ビットのみであり、それ以外のビットデータは不要となる。よってこの際、必要となるビットを特定するためには、レジスタ４１に対してビットマスク処理を行い、必要なビットを取り出さなくてはいけない。次に、処理ブロック中の次のデータ４乃至７の最上位ビットをロードするために、メモリ４０のアドレス４からロード処理を行い、さらに、処理ブロック中の残りのビットをロードするために、メモリ４０のアドレス８からのロード処理とアドレス１２からのロード処理を行って、処理ブロック全体として、４回のロード処理が必要になる。

【００１０】また、最上位ビットのビットプレーンの処理が終わり、次のビットプレーンの処理に進む場合も、最上位ビットのビットプレーンの処理と全く同じように４回のロード処理を行い、レジスタ４１に対してビットマスク処理を行うことで必要なビットを取り出す。この処理が全てのビットプレーン（図３では８枚）に対して、繰り返される。

【００１１】

【発明が解決しようとする課題】以上の説明による従来の符号化処理は、符号化プロセッサに多くのロード命令とビットマスク処理を要求し、ビットプレーン符号化処理の効率化・高速化を図る際の障害となる。又、以上の説明による処理をソフトウェアにより実装しても又、その結果は同じである。

【００１２】一方、前述の特開平０８－１４９３０８号公報や特開平０９－０２７７５２号公報では、ビットプレーン符号化の論理的なアルゴリズムについては述べているが、その具体的な実装方法については述べていない。

【００１３】本発明は以上の問題点に対して鑑みてなされたものであり、ビットデータ、特に多値データのビットプレーンの転送の高速化、効率化を行うことを目的とする。

【００１４】

【課題を解決するための手段】本発明の目的を達成するために、例えば本発明のデータ処理装置は以下の構成を備える。すなわち、所定のメモリ間でデータ転送を行うデータ処理装置であって、複数のビットにより構成される多値データが第１のメモリに複数存在し、当該複数の多値データから同一ビットプレーンに属するビットからなるビットプレーンデータ群を、当該ビットプレーンデータ群単位で第２のメモリにデータ転送する。

【００１５】本発明の目的を達成するために、例えば本発明のデータ処理装置は以下の構成を備える。すなわち、所定のメモリ間でデータ転送を行うデータ処理装置であって、複数のビットプレーンデータ群が存在し、各ビットプレーンデータ群において同一多値データに属するビットデータを所定の順番で選択することで、多値データを生成する。

【００１６】本発明の目的を達成するために、例えば本発明のプロセッサは以下の構成を備える。すなわち、所定のメモリから所定のレジスタにデータをロードするプロセッサであって、複数のビットにより構成される多値データが前記所定のメモリに複数存在し、当該複数の多値データから同一ビットプレーンに属するビットからなるビットプレーンデータ群を当該ビットプレーンデータ群単位で前記所定のレジスタにロードする。

【００１７】

【発明の実施の形態】以下添付図面に従って、本発明を好適な実施形態に従って詳細に説明する。

【００１８】［第１の実施形態］本実施形態では、多値データービットプレーンデータ変換機能を備えるＤＭＡ回路を用いて、ビットプレーン符号化を行うプロセッサである符号化プロセッサが入力データ（ビットプレーン符号化の対象データとなる多値データ）にアクセスする前に、当該入力データを格納するメモリ上で、多値データからビットプレーンデータの変換を行っておく。

【００１９】図５に、本実施形態におけるＤＭＡ回路が行う処理について示す。メモリ５０上の多値データの並びは、図４に示した多値データの並びと全く同一である。本実施形態におけるＤＭＡ回路は、メモリ５０上の他の領域５１（必ずしもメモリ５０と同一の物理メモリである必要はなく、他のメモリであってもよい）に転送し、その転送の際に、多値データ形式からビットプレーンデータへの変換を行う。

【００２０】メモリ領域５１では、各多値データ（図５では、データ０乃至データ１５）の最上位ビット（ビット７）が各多値データ順（例えばデータ０，データ１，データ２、、、の順）に集められて、斜線部で示されるように１箇所にまとめて格納される。ビット６についても、同様に格納される。このように、メモリに各ビット

を格納することで、符号化プロセッサは１回のロード処理で、要求するビットプレーンの全データ（ここでは２ビットプレーン分（ビット７，６）の全データ）が獲得できる。このため、図４において問題となった、符号化プロセッサの処理のオーバヘッドを減らすことができる。また、レジスタ５２に示すように、要求するビットプレーンのデータがひと固まりの形で得られるため、同一ビットプレーンにおける近傍画素の係数値を一度に得られ、ＪＰＥＧ２０００などで用いられる近傍画素の係数値を用いたエントロピー符号化を効率的に実装できる。

【００２１】また、以上のＤＭＡ回路を導入して、符号化プロセッサの負荷が減っても、ＤＭＡ転送のオーバヘッドが追加されるが、符号化プロセッサが他に行っている処理とＤＭＡ転送をうまくオーバーラップさせることによって、システム（例えば、ビットプレーン符号化を用いた画像符号化システム）全体としての処理を効率化・高速化できる。その際、このＤＭＡ回路は、同システムにおいて、メモリ領域間のＤＭＡ転送を行うハードウェアとして実施される。

【００２２】図１に、本実施形態におけるＤＭＡ回路の概略構成と共に、同回路を含む符号化システムの概略構成を示す。

【００２３】図１において、メインメモリ１７上にビットプレーン符号化の際、入力データとなる多値データが、図４、図５で示した例と同じ形式で格納されており、この多値データを、ＤＭＡ回路１０がデータ変換を行いながらメインメモリ１７から符号化プロセッサ１９にとってのローカルメモリ１８に、ビットプレーンデータとしてＤＭＡ転送する。ＤＭＡデータ転送は、バス１６を介して行われるが、同図では単純化して表現されており、これは必ずしも単一のバスであるとは限らない。（メインメモリと、ローカルメモリ・符号化プロセッサが別のバスにあるような場合も想定できる。また、逆にメインメモリしか存在せず、１７と１８が、ともにメインメモリ上にある場合も想定できる。）

メインメモリ１７からローカルメモリ１８へのデータ変換のなされ方は、図５に示した例と同じであり、符号化プロセッサ１９は、ローカルメモリ１８に格納されたビットプレーン単位のデータ（ビットプレーンデータ）をロードし、ビットプレーン符号化を行うことができる。

【００２４】ＤＭＡ回路１０の内部構成は同図に示すように、転送元アドレス制御回路１１、ビット選択書き込み回路１２、ワードバッファ１３、転送先アドレス制御回路１４、バスインタフェース回路１５の５つの部分回路からなる。

【００２５】転送元アドレス制御回路１１は、ＤＭＡ転送の転送元アドレスを制御する回路であり、転送先への１回分のデータをワードバッファ１３に満たすため、通常、複数回のロードをメインメモリ１７に対して発行する。図１の例において、ローカルメモリ１８のアドレス（ｙ＋０）に格納するデータをワードバッファ１３に満たすためには、メインメモリ１７のアドレス（ｘ＋０），（ｘ＋４），（ｘ＋８），（ｘ＋１２）の４ワード分のデータが必要であるため、メインメモリ１７に対し、４回のロードを発行する。

【００２６】ビット選択書き込み回路１２は、転送元アドレス制御回路１１により得られた転送元データ幅分のデータから、必要なビットプレーンに相当するビットを選択して取り出し、選択されたビットの値をワードバッファ１３に書き込む回路である。

【００２７】具体的に、ビット選択書き込み回路１２が行う処理について、ローカルメモリ１８のアドレス（ｙ＋０）に格納するデータをワードバッファ１３に満たす場合を例に説明する。まず、メインメモリ１７のアドレス（ｘ＋０）のデータ（データ０乃至３）が読み込まれ、そのうち、ローカルメモリ１８のアドレス（ｙ＋０）に格納されるビットプレーン、すなわち、各多値データ（データ０乃至３）のビット７とビット６がビット選択書き込み回路１２により取り出され、ワードバッファ１３に各多値データのビット７、及びビット６の夫々を集めて格納する。次に、メインメモリ１７のアドレス（ｘ＋４）のデータが読み込まれ、ビット選択書き込み回路１２により各多値データのビット７とビット６が取り出され、ワードバッファ１３に各多値データのビット７，及びビット６の夫々を集めて、先に格納されたアドレス（ｘ＋０）からロードされたビット群に隣接して格納する。メインメモリのアドレス（ｘ＋８）、（ｘ＋１２）に対しても同じように処理が繰り返され、メインメモリ１７のアドレス（ｘ＋１２）に対する処理が終わった時点で、ワードバッファ１３には、ローカルメモリ１８のアドレス（ｙ＋０）に書き込む値が格納されている。

【００２８】ワードバッファ１３は、転送先（ローカルメモリ１８）のデータバス幅を持ったデータバッファで、転送元（メインメモリ１７）からロードを行うことによって集めたビットを格納し、転送元からの複数回のロードが完了した時点で、転送先（ローカルメモリ１８）へ格納するデータを保持している。

【００２９】転送先アドレス制御回路１４は、ワードバッファ１３が格納するデータの転送先のアドレスを制御する回路である。転送元アドレス制御回路１１が、アドレス（ｘ＋０）、（ｘ＋４）、（ｘ＋８）、（ｘ＋１２）へ４回のロードを発行し、ワードバッファ１３に転送先（ローカルメモリ１８）へ格納するデータの保持が完了した後に、転送先アドレス制御回路１４は、アドレス（ｙ＋０）への書き込みを発行する。

【００３０】バスインタフェース回路１５は、ＤＭＡ回路１０のバス１６へのアクセスタイミングを制御する回路であり、転送元アドレス制御回路１１、転送先アドレ

ス制御回路１４からの要求に従って、実際のデータ転送をバス上で実行する。

【００３１】図１では、図３、図４において仮定したパラメータ（メモリバス幅３２ビット、多値データのビット幅８ビットなど）と同じパラメータを用いた際の例を示してあるが、転送元アドレス制御回路１１の制御方法や、ビット選択書き込み回路１２のビット選択の仕方、ローカルメモリ１８へのビットの書き込み方などを可変パラメータとして設定可能することによって、様々なメモリバス幅や多値データのデータ形式、多値データの処理単位（処理ブロックの大きさ）に対応できるように、ＤＭＡ回路１０を構成することも可能である。

【００３２】本実施形態におけるＤＭＡ回路及びその方法は、以上の説明の通り、メインメモリに格納されている各多値データに対して、当該多値データを構成する各ビット毎にビット群データを生成し、このビット群データ毎にローカルメモリに格納することで、１回のロード処理で、要求するビットプレーンの全データを獲得することができ、ビットプレーン符号化処理を簡略化することができると共に、同処理に要する処理時間の短縮化が可能となる。

【００３３】［第２の実施形態］第１の実施形態で用いたＤＭＡ回路は、ビットプレーン符号化処理の際に用いたが、逆方向のビットプレーン復号化処理の際にも、当該ＤＭＡ回路を用いることができる。本実施形態では、このＤＭＡ回路を用いた復号システムにおける復号処理について説明する。

【００３４】一般にビットプレーン復号化処理は、図２に示したデータの流れを逆方向にたどって処理することになる。すなわち、符号化データを、まず、ビットプレーン復号化によって、ビットプレーンデータを生成する。この復号化されたビットプレーンデータは、ビットプレーン単位で生成されているため、多値データ配列に戻す必要がある。多値データ配列に戻った後に（必要ならば逆量子化と）逆変換符号化を行い、もとの画素値が再現される。この一連の復号化の流れにおいて、生成されたビットプレーンデータから多値データを生成する際に、前記ＤＭＡ回路を用いることができる。

【００３５】図６に示すようにビットプレーン復号化直後のデータはビットプレーン単位で、メモリ６０内に同図の通り格納されている。図６は、本実施形態におけるＤＭＡ回路が、ビットプレーンデータから多値データ生成の際に行う処理を示す図である。

【００３６】メモリ６０に格納されたビットプレーン毎のビットプレーンデータに基づいて、各ビットプレーンの特定の位置に相当するビットを、多値データのビット深さ分集めてくることにより、メモリ６１内に同図の通り、多値データ配列に並べ替えることができる。

【００３７】同図の例で、復号化時のＤＭＡ回路の変換動作を説明する。メモリ６０のアドレス（ｙ＋０）には、ビット（プレーン）７，６のデータが存在しており、ＤＭＡ回路はこのビット７，６をロードする。転送先であるメモリ６１のアドレス（ｘ＋０）には、４個分の多値データ（データ０乃至３）を集めて格納するため、アドレス（ｙ＋０）からロードした前述のデータのうち、斜線部で示された多値データ０乃至３に相当する部分（ビット７ではビット番号３１～２８、ビット６ではビット番号１５～１２）をビット選択書き込み回路１２で取り出して、各多値データ毎にビットを集めてワードバッファ１３に書き込む。この動作を、アドレス（ｙ＋４）、（ｙ＋８）、（ｙ＋１２）に対しても繰り返し行う。その結果、バッファ１３には各データ０，１，２，３の上位ビットから読み込まれていき、最後には各データ０，１，２，３がバッファ内で復元されることになる。

【００３８】アドレス（ｙ＋１２）に対する処理が終了した時点で、メモリ領域６１のアドレス（ｘ＋０）に格納するデータがワードバッファ１３内に格納されているので、ワードバッファ１３に格納されたデータ（データ０乃至３）のデータをアドレス（ｘ＋０）に格納する。以下、メモリ領域６１のアドレス（ｘ＋４）、（ｘ＋８）、（ｘ＋１２）に対しても同様の処理を繰り返し、最終的に、４×４の多値データ配列を生成する。

【００３９】本実施形態におけるＤＭＡ回路及びその方法は、以上の説明により、ビットプレーンデータから多値データを復元することができる。

【００４０】［第３の実施形態］本実施形態では、複数の多値データを格納したメモリから、各ビットプレーンに含まれるビットをレジスタにロードする際の処理について説明する。

【００４１】又、本実施形態では、第１の実施形態においてビット選択書き込み回路１２が行う、転送元アドレス制御回路１１により得られた転送元データ幅分のデータから、必要なビットプレーンに相当するビットを選択して取り出す処理を、レジスタにロードする処理も含めて、ビット選択ロード命令としてプロセッサ（例えばＣＰＵや、ＤＳＰなど）の機能に含め、当該プロセッサにより、メモリからの多値データのロードと同時に、必要なビットプレーンのデータを選択してレジスタにロードする。もちろん、他にも上述のレジスタへのビットプレーンデータのロード処理は、第１の実施形態におけるＤＭＡ回路で行っても良いことは明白である。

【００４２】図３，４において仮定したパラメータと同じパラメータ（メモリバス幅３２ビット、多値データのビット幅８ビットなど）を用いた場合の、前記ビット選択ロード命令を用いた場合のデータ処理の様子を図７に示す。メモリ７０，７１，７２，７３上の多値データの並びは、第１，２の実施形態の場合と全く同じである。図７でも、第１，２の実施形態と同じく、最上位のビットプレーンに着目し、最上位のビットプレーンのビット

を斜線部で示す。

【００４３】最上位のビットプレーンを指定して、１回目のビット選択ロード命令を実行すると、メモリ７０に対して１ワード分のデータを読み出した後、最上位のビットプレーンのビットだけを選択して集め、レジスタ７４のビット番号３１，３０，２９，２８で特定されるビット位置に書き込む。同様に、２回目のビット選択ロード命令を、メモリ７１とレジスタ７４に対して実行する。ただし、今度はレジスタ７４へのビットプレーンのビットの書き込み位置を１回目とずらしてビット番号２７で特定されるビット位置からと指定することによって、メモリ７１から読み込まれたビットプレーンのビットのうち、最上位のビットプレーンのビットが、レジスタ７４のビット番号２７，２６，２５，２４で特定されるビット位置に書き込まれる。以下、同様に、３回目、４回目のビット選択ロード命令を、メモリアドレスを変えながら同一のレジスタに対して実行することによって、最終的には、最上位のビットプレーンに対応する全１６ビットのデータが、ひとつのレジスタ上に固まって配置されることになる。

【００４４】結果として、通常のロード命令を使用した場合と、上述のビット選択ロード命令を使用した場合で、メモリ（７０，７１、７２、７３）からの読み込みの回数は同じであるが、メモリ（７０，７１、７２、７３）から読み出されたデータは次のステップではビットプレーン符号化が施されるので、ビットプレーン毎にレジスタにロードすることができるビット選択ロード命令を使用したほうが、プロセッサ上のレジスタ内のビットの使用効率が高く、ビットプレーンデータ格納のためのレジスタ消費が少なくて済む。また、通常のロード命令では、ロード命令実行後のレジスタにおいて、着目するビットプレーンのビットが散在し、煩雑なビットマスク処理が必要となるが、ビット選択ロード命令使用後のレジスタは、図５のレジスタ５２のように、着目するビットプレーンのビットが固まって配置されているため、ビットマスク処理も容易である。また、あるいは、ビットマスク処理を使用せずに、シフト演算だけで必要なビットをキャリービットとして取り出すこともできる。

【００４５】図８に、本実施形態におけるビット選択ロード命令を実行することで、メモリからレジスタに読み込まれるビットの様子を示す。同図において、メモリ８０上に、各ｄビットの多値データが連続して存在しており、プロセッサのワード幅をｗビットとする。

【００４６】ビット選択ロード命令は、まず、メモリ８０の指定されたアドレス位置から１ワード分のデータｗビットを読み込む。その後、このｗビットの中に含まれる多値データの中から各ｎ＝１ビットを選択し、全部でｗ／ｄ×ｎ＝ｗ／ｄ×１＝ｗ／ｄビットを集めて、レジスタ８１へ書き込みを行う。図８では、選択されたビットを斜線部で示す。

【００４７】レジスタ８１で、選択されたビットの書き込みが行われるレジスタ８１の部分以外の値は、ビット選択ロード命令の実行によって変化せず、以前の値を保持している。

【００４８】メモリ８０上において、各ｄビットの多値データの中から、どのビットプレーンを選択するかは、ビット読み出し位置ｘ（０≦ｘ＜ｄ）によって指定する。また、レジスタ８１において、選択されたビットをどのビット位置に書き込むかは、ビット書き込み位置ｙ（０≦ｙ＜ｗ）によって選択する。

【００４９】ｘおよびｙは、ビット選択ロード命令へのオペランドとして与えられ、例えば、ビット選択ロード命令のニモニックをＢＬＯＡＤとすると、

ＢＬＯＡＤ　ｄｒ，［ｓｒ＋ｏｆｆｓｅｔ］，＃ｘ，＃ｙ

のような形式でアセンブラ上で記述できる。ここで、ｓｒは、メモリ８０上のロードアドレスを指定するアドレスベースレジスタであり、ｓｒの内容とｏｆｆｓｅｔの和がロードアドレスとなる。

【００５０】ｄｒは、ロード先のレジスタを示している。＃ｘはメモリ上のビット読み出し位置を指定する即値オペランド、＃ｙはレジスタへのビット書き込み位置を指定する即値オペランドである。ｘ，ｙを即値オペランドで指定しているのは、あくまで一例であり、レジスタオペランドなど他のアドレッシング方式を用いてもよい。

【００５１】上記のアセンブラ記述方式を使って、図７を用いて説明した処理例のアセンブラコードを記述すると、次のようになる。ここでは、ｒ１はメモリ８０上のロードアドレスを格納しており、ｒ０（レジスタ）にロード結果が入る。

【００５２】

ＢＬＯＡＤ　ｒ０，［ｒ１］，＃０，＃０
ＢＬＯＡＤ　ｒ０，［ｒ１＋４］，＃０，＃４
ＢＬＯＡＤ　ｒ０，［ｒ１＋８］，＃０，＃８
ＢＬＯＡＤ　ｒ０，［ｒ１＋１２］，＃０，＃１２

ここでは、ワード幅ｗビット、多値データのビット幅ｄビットは決まったものとして与えたが、これらの値も可変にできるよう命令を定義、実装することは可能であり、その場合、ｗやｄは、ビット選択ロード命令のオペランドとして与えるか、あるいは、異なったｗやｄの値に対応した複数のビット選択ロード命令を定義して使用することができる。

【００５３】また、本実施形態では、一つのビットプレーンの取り出しのために、ｄビットの多値データのうち、ｎ＝１ビットを選択する場合について示してあるが、各多値データ中から複数のビット（ｎ＞１）を選択する場合にも、容易に拡張可能であることは明白である。

【００５４】また、図９のフローチャートを用いてBLOA

D命令の処理フローを説明する。図９は、前述の説明で、
ＢＬＯＡＤ　ｄｒ，［ｓｒ＋ｏｆｆｓｅｔ］，＃ｘ，＃ｙ
という命令を実行した際の、処理フローの説明である（ｎ＝１と仮定）。

【００５５】まず、ステップＳ９０では、sr+offsetの計算を行った上で、メモリアドレスsr+offsetから1ワード分のデータの読み出しを行う。

【００５６】次に、ステップＳ９１からステップＳ９５にかけては、1ワード分のデータ中から必要なビットを取り出す処理を行う。ステップＳ９１は、ビット取り出し処理のループの開始ステップ、ステップＳ９５はループの終了ステップであり、ループは、1ワードのビット幅（wビット）を多値データのビット幅(dビット)で割った回数だけ繰り返す。

【００５７】ループ中の最初のステップＳ９２では、メモリから読み出した1ワードの中から、多値データ1個に相当するｄビットを取り出す。次に、ステップＳ９３では、そのｄビットの中からＢＬＯＡＤ命令のオペランドとして指定されたビット位置#xで指定されたビットをｄ対１のビットセレクタを使用して選択する。ここでは、仮に、ビット位置#xはｄビットの上位から数えているため、ビット位置（ｄ－１－ｘ）の1ビットを取り出す。次のステップＳ９４では、ステップＳ９３で取り出した1ビットを、転送先のレジスタｄｒのビット位置（３１－＃ｙ）から順に、ループ毎に下位（ビット位置０が最下位）にずらしながら書き込みを行う。すなわち、最初のループでは、レジスタｄｒのビット位置（３１－＃ｙ）に書き込み、次のループでは、ビット位置（３１－＃ｙ－１）に書き込み、以下、ループごとに書き込むビット位置を下位方向にずらして書き込みを行う。

【００５８】ループの処理が既定回数終わった時点で、レジスタｄｒへの選択ビットの書き込みは終了し、BLOAD命令は終了する。

【００５９】なお、このフローチャートでは、ステップＳ９１からステップＳ９５にかけて、１ループで１ビットずつ取り出し、書き込む説明になっているが、これは論理的な説明であり、ハードウェアによる実装上は、ステップＳ９３のビットセレクタを複数個設けることにより、一度に複数ビットを取り出して、書き込むことが可能である。もし、ビットセレクタをw/d個設け、並列に動作させれば、ループ内の処理は、1度で済むことになる。

【００６０】［第４の実施形態］すべての上述の実施形態の動作説明を補足するため、次に、JPEG2000による符号化処理について説明する。

【００６１】図10は、JPEG2000による符号化処理の全体のフローを示した図である。最初のステップＳ１００はDCレベルシフト処理であり、入力画像の画素値が符号無しの値で表現される場合に、画素値から定数を引いて、符号つきの値に変換する。入力画像の画素値が符号ありの値で表現される場合は、このステップでは何も行わない。

【００６２】次のステップＳ１０１はコンポーネント変換処理であり、３色の色コンポーネントの値を３次元の定数行列乗算処理により変換する。RGB形式の色表現を、YUV形式の色表現に変換する処理と同等の処理であり、色コンポーネントの値の分布を変えることで効率的な圧縮を可能とする。ステップＳ１０１は、オプションであるため、符号化処理に際して行っても行わなくても良い。

【００６３】次に、ステップＳ１０２では、画素値に対して二次元の離散ウェーブレット変換（ＤＷＴ）を行う。まず、一次元のDWTについて説明すると、JPEG2000では、5-3 Reversible filterと、9-7 Ireversible filterの2種類のフィルタが定義されており、どちらかを使用して変換を行う。ここでは、簡単のため、5-3 Reversible filterについて説明する。一次元DWTへの入力値をＸ（ｎ）（ｎ＝０，１，…，Ｎ－１）とし、出力値を、Ｙ（ｎ）（ｎ＝０，１，…，Ｎ－１）とすると、5-3 Reversible filterの処理は次の2段階の数式で表される。

【００６４】
Ｙ（２ｎ＋１）＝Ｘ（２ｎ＋１）－（Ｘ（２ｎ）＋Ｘ（２ｎ＋２））／２
Ｙ（２ｎ）＝Ｘ（２ｎ）－（Ｙ（２ｎ－１）＋Ｙ（２ｎ＋１）＋２）／４
ここで、［ｘ］はｘを超えない最大の整数を表すものとし、入力値Ｘ（ｎ）のｎの範囲がもともとの範囲０～（Ｎ－１）を超える場合は、規格で定めた方法でＸ（ｎ）の定義される範囲を拡張してから使用する。Ｙ（２ｎ）は低周波（Ｌ）成分の係数、Ｙ（２ｎ＋１）は高周波（Ｈ）成分の係数に相当する。

【００６５】9-7 Ireversible filterの場合は、数式が６段階に増え、数式中の定数係数の値も変わって複雑になるが、各段階における基本的な計算方法は同様である。いずれのフィルタを使用しても、Ｎ個の入力値から、Ｎ個の出力値が生成され、出力値のうち、偶数番目のＮ／２個は低周波（Ｌ）成分、奇数番目のＮ／２個は高周波（Ｈ）成分である。

【００６６】次に、二次元DWTの処理について図１１を用いて説明する。図１１において、二次元の入力画素（群）１１０に対しては、まず、各列毎に、垂直方向の一次元DWT処理を行い、一次元DWTの低周波（Ｌ）側の出力と高周波（Ｈ）側の出力を集めることで、垂直ＤＷＴ後の係数値１１１となる。次に、係数値１１１の各行毎に、水平方向の一次元DWT処理を行い、同様に、L側、H側の出力係数を集めることで、二次元DWT後の出力係数値１１２が得られる。１１２において、水平・垂直ともL側の成分を集めた部分をLLサブバンド、垂直方向はL成

分、水平方向はH成分を集めた部分をHLサブバンド、垂直方向はH成分、水平方向はL成分を集めた部分をLHサブバンド、水平・垂直ともH側の成分を集めた部分をHHサブバンドと呼ぶ。LLサブバンドに関しては、通常、さらに二次元DWTを繰り返し施すことで、より細かなサブバンドに分割する。

【００６７】ステップＳ１０３では、分割された各サブバンドの係数の量子化処理を行う。入力係数値をx、量子化ステップをΔ、とすると量子化後の係数値ｑは、

ｑ＝ｓｉｇｎ（ｘ）×［ａｂｓ（ｘ）／Δ］

となる。ここで、ｓｉｇｎ（ｘ）はｘの符号で、ｘ＜０のとき－１，ｘ≧０のとき１を返し、ａｂｓ（ｘ）はｘの絶対値を返す。［ｘ］はｘを超えない最大の整数である。量子化ステップΔは、各サブバンド毎に一定の値である。

【００６８】更にステップＳ１０３までは、画素値や係数値は計算の都合上、２の補数形式で表されることが多いが、ステップＳ１０３から次のステップＳ１０４にかけては、係数の符号と絶対値を別々に分けて処理する必要があるため、係数値の表現形式を、係数の符号を示す１ビットと絶対値を示す残りのビットで表現する、符号・絶対値形式に変換する。

【００６９】次のステップＳ１０４では係数ビットモデリング処理を行うが、ステップＳ１０２，Ｓ１０３の処理がサブバンドに対する処理であるのに対して、ステップＳ１０４の処理はサブバンドをさらに分割してコードブロックという単位で処理を行う。

【００７０】図１２は二次元DWT後の画像係数データ１２０とサブバンド１２１、コードブロック１２２の関係を示した図で、二次元DWT（ステップＳ１０２）後の係数データのうち、一つのサブバンド１２１が取り出されて量子化（ステップＳ１０３）され、そのサブバンド１２１がさらに点線のように分割されてそのひとつずつがコードブロック１２２として、ステップＳ１０４で処理される。

【００７１】また図１０において、ステップＳ１０３までは、画素値や係数値を多値のデータとして扱うのに対して、ステップＳ１０４では、多値データをビットプレーンに分解して、ビットプレーンデータとして処理をするため、ステップＳ１０４のデータ入力の部分で、上述の実施形態が適用される。

【００７２】ステップＳ１０４のデータ入力の処理順序に関して、図１３を用いて説明する。図１３左側の図は、コードブロックの多値データ（符号・絶対値形式）を示した図で、図における横方向・奥行き方向がコードブロックの空間上の広がりを示し、図における高さ方向が多値データのビット深さを示している。１３０は、係数の符号ビットだけからなるビットプレーンであり、１３１は、係数の絶対値を示すビットで、点線により、ビット深さごとのビットプレーンに分割される。絶対値ビットは、図中上から（MSBから）下に（LSBに）向かって並んでいるものとすると、絶対値ビットプレーン１３１はMSBからLSBに向かって順に処理される。符号ビットプレーン１３０は、絶対値ビットプレーン１３１の処理が進む途中で処理のために参照され、条件によって、符号ビットも処理される。

【００７３】図１３右側の図は、絶対値ビットプレーン１３１から一つのビットプレーン１３２を取り出したときに、そのビットプレーン内でのビットの処理順序を示している。ビットプレーン１３２は、縦４ビット毎に分割され、右側の図の矢印の向きに従って処理される。縦方向の小さな矢印が縦4ビットを示し、縦4ビットの処理が一度終わるごとに、右隣の４ビットを上から下に処理する。処理が右端に到達すると、次の行の縦４ビットを処理し、以下、全コードブロックに対応するビットプレーンデータが処理されるまで、このスキャンは、一つのビットプレーン内で３回行われる。３回のスキャンが行われる理由は、３回の各スキャンで条件によって処理対象となるビットとならないビットがあり、３回スキャンすることで、全ビットの処理が完了するためである。各スキャンにおける対象ビットの選択条件については、規格上規定されている。

【００７４】ステップＳ１０４では、図１３に示した順序で係数ビットをスキャンしながら、次の算術符号化処理（ステップＳ１０５）で使用するコンテキストおよび符号化されるビット値の計算を行う。この計算には、図１４に示すように符号化対象となる係数ビット１４０、およびその周囲８近傍の係数ビット１４１（斜線部）のビット値と、それに付随して計算される状態が使用される。図１４において、中央の４行が現在処理中の縦４ビットの並びとすると、１４２は一つ前の縦４ビットに属し、１４３は一つ次の縦４ビットに属しているが、対象画素の周囲８近傍が必要となることから、現在処理中の縦４ビットの処理にも１４２、１４３の係数ビットの値、状態情報が必要である。コンテキストおよび符号化されるビット値はこれら多くのビット情報を参照した上で、計算され、次の算術符号化ステップＳ１０５に渡される。

【００７５】図１０において、ステップＳ１０５の算術符号化処理では、ステップＳ１０４で計算されたコンテキストと符号化されるビット値を入力として、コンテキストベースの算術符号化を行い、係数データのデータ量を圧縮する。

【００７６】次のステップＳ１０６では、ステップＳ１０５で圧縮されたデータを、適切な単位でパケット化し、必要なヘッダセグメントを付加して、出力ビットストリームの生成を行う。

【００７７】以上の処理によって、JPEG2000の符号化処理は終了する。なお復号化処理は、図１０の手順を逆にたどり、各ステップの逆変換を行うことで実現できる。

図１０で示したJPEG2000の符号化処理の手順の中で、上述の実施形態が適用されるのはステップＳ１０４の係数ビットモデリング処理である。本ステップから処理対象のデータが多値データからビットプレーンデータへ変わるため、上述の実施形態による変換処理を行うことで、ステップＳ１０４の処理を効率的に実行できる。

【００７８】図１５に、ステップ１０４に上述の実施形態を適用した場合の例を示す。図１５において、ステップＳ１０４における処理対象が４ｘ４の多値係数データ１５０であるとする。説明を簡単にするため、係数データのサイズを４ｘ４としたが、実際にはより大きなサイズとなる。係数データ１５０は図１３で説明したように、図中の番号の順（０，１，２，…，１５）で、上位ビットプレーンから下位ビットプレーンに向かって処理される。ここで、この係数データがメモリ（例えばメインメモリ１７）上にある時点での状態を、１５１、１５２，１５３，１５４に示す。１５１，１５２，１５３，１５４はそれぞれメモリ上で１ワードの記憶領域であり、それぞれに係数０～３、係数４～７、係数８～１１、係数１２～１５と、４係数ずつのデータが含まれている。各多値係数データの最上ビットが符号ビットであるとすると、図中、斜線部が符号ビットに相当し、各メモリワードの中に分散して存在していることがわかる。これを、上述の第１の実施形態もしくは第３の実施形態により、各ワードから符号ビットの位置にあるビットを集めてくる処理を行えば、結果として、ローカルメモリあるいはレジスタ上において、符号ビットだけが集まったデータ１５５を得ることができる。

【００７９】一度、データ１５５の形で符号ビットを集めることができれば、以後、係数０～１５のどの符号ビットの値が必要な場合も、ビットマスクとデータ１５５の論理積（ＡＮＤ）を計算するだけで容易に得ることができ、図１４に示したように、符号化対象ビットの近傍画素のデータの値を得る処理を効率化できる。図１３左の図で示したように、特に、符号ビットプレーン１３０は、絶対値ビットプレーン１３１の処理中ずっと必要になるため、図１５のデータ１５５のように格納しておくことで、データ１５５のうちの任意のビットを効率的に参照することができる。

【００８０】

＊【発明の効果】以上の説明により本発明によれば、ビットデータ、特に多値データのビットプレーンの転送の高速化、効率化を行うことができる。

【図面の簡単な説明】

【図１】本発明の第１の実施形態におけるＤＭＡ回路の概略構成と共に、同回路を含む符号化システムの概略構成を示す。

【図２】一般化された画像のビットプレーン符号化方式の処理の流れを示す図である。

【図３】４×４の４ビット係数値の配列３０の４つのビットプレーン３１，３２，３３，３４に分解する処理の例を示す図である。

【図４】従来の符号化処理において、符号化プロセッサのレジスタ幅（１ワード）＝３２ビット、メモリデータバス幅Ｗ＝３２ビット、多値データのビット幅ｄ＝８ビットとした場合の例を示す図である。

【図５】本発明の第１の実施形態におけるＤＭＡ回路が行う処理を示す図である。

【図６】本発明の第２の実施形態におけるＤＭＡ回路が、ビットプレーンデータから多値データ生成の際に行う処理を示す図である。

【図７】ビット選択ロード命令を用いた場合のデータ処理の様子を示す図である。

【図８】ビット選択ロード命令を実行することで、メモリからレジスタに読み込まれるビットの様子を示す図である。

【図９】ＢＬＯＡＤ命令の処理フローチャートである。

【図１０】ＪＰＥＧ２０００における符号化処理のフローチャートである。

【図１１】２次元ＤＷＴの処理を説明する図である。

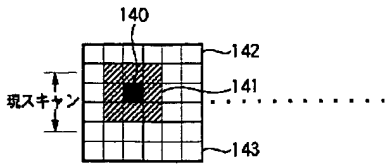
【図１２】２次元ＤＷＴ後の画像係数データ２１０とサブバンド１２１，コードブロック１２２の関係を示す図である。

【図１３】ステップＳ１０４のデータ入力の処理順序に関して説明する図である。

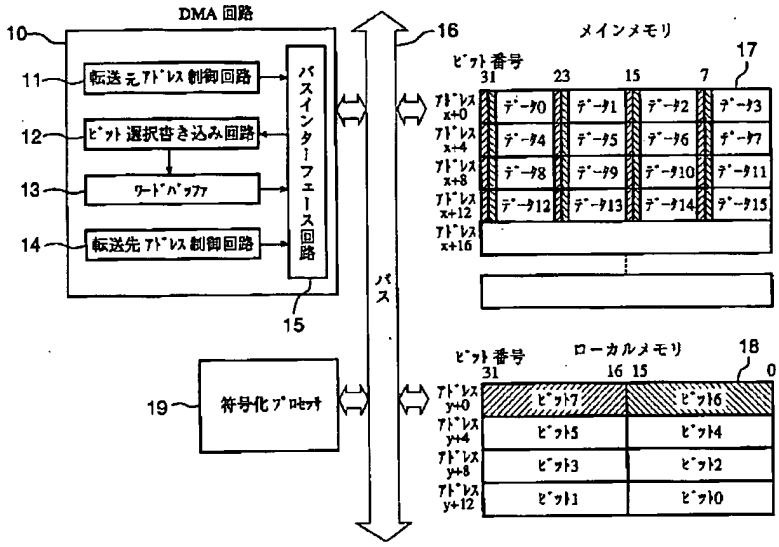
【図１４】符号化対象となる係数ビット１４０、およびその周囲８近傍の係数ビット１４１（斜線部）を示す図である。

【図１５】ステップＳ１０４に本発明の実施形態を適用した場合の例を示す図である。

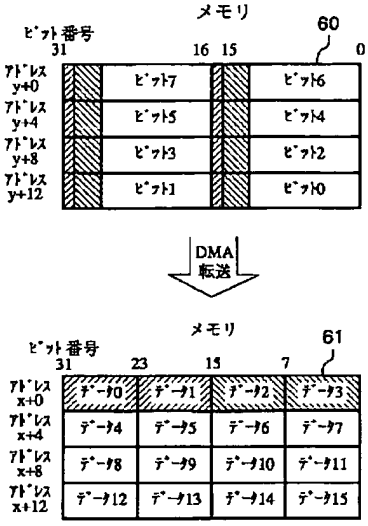
【図１４】



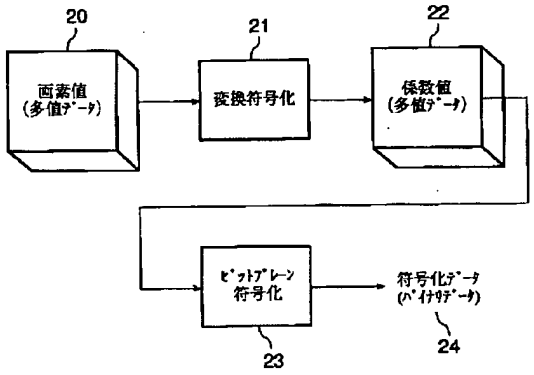
【図１】



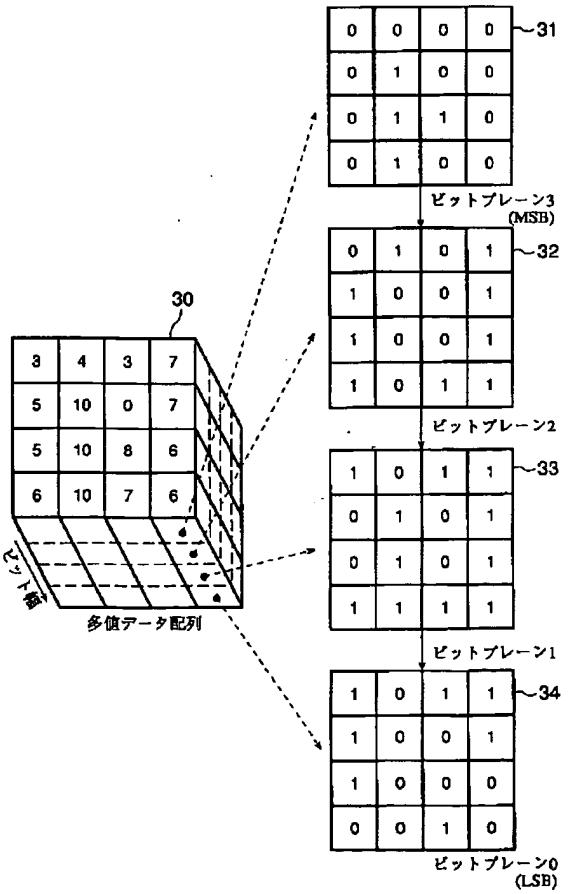
【図６】



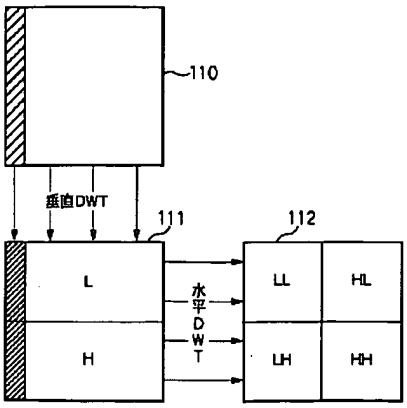
【図２】



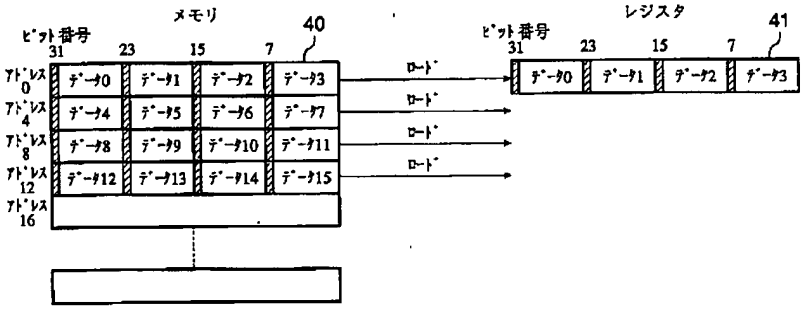
【図３】



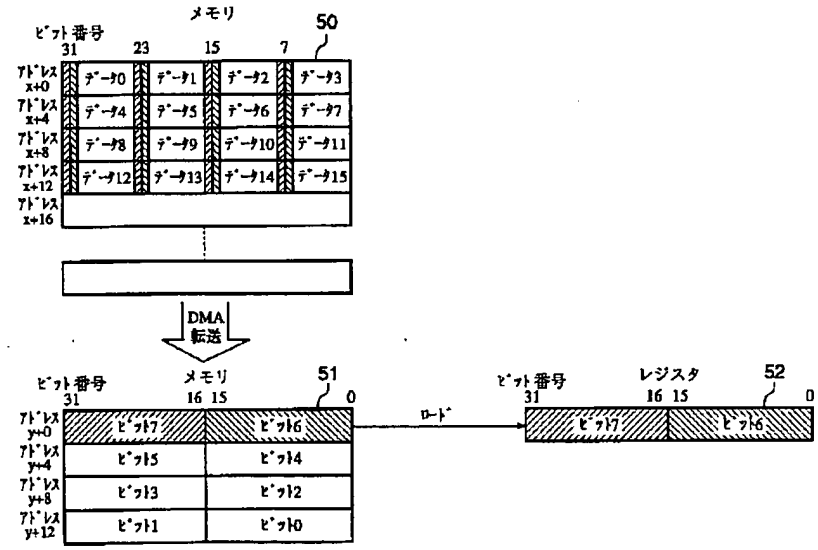
【図１１】



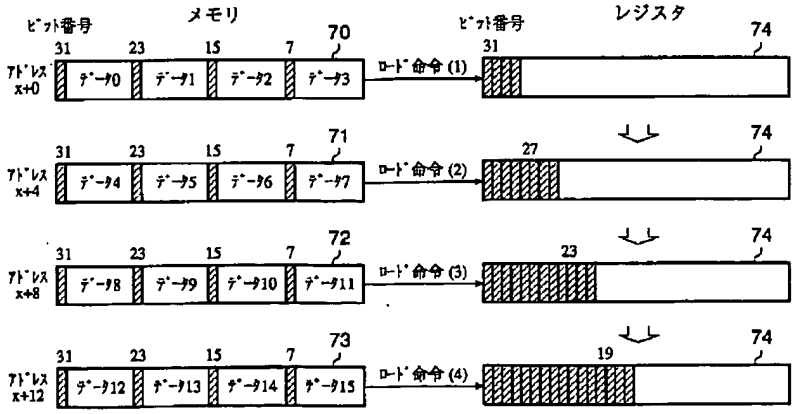
【図４】


メモリ
40
ビット番号
31 23 15 7
アドレス 0
アドレス 4
アドレス 8
アドレス 12
アドレス 16
データ0 データ1 データ2 データ3
データ4 データ5 データ6 データ7
データ8 データ9 データ10 データ11
データ12 データ13 データ14 データ15
ロード
ロード
ロード
ロード
レジスタ
41
ビット番号
31 23 15 7
データ0 データ1 データ2 データ3

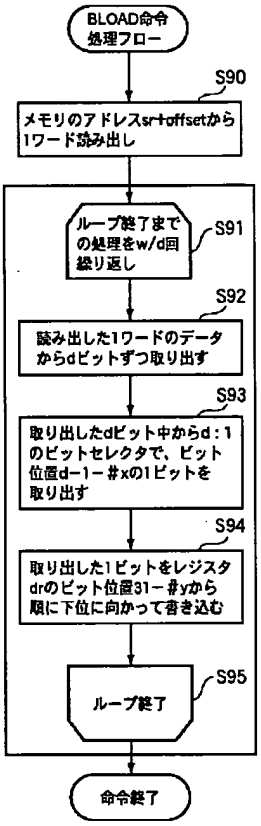
【図５】


メモリ
50
ビット番号
31 23 15 7
アドレス x+0
アドレス x+4
アドレス x+8
アドレス x+12
アドレス x+16
データ0 データ1 データ2 データ3
データ4 データ5 データ6 データ7
データ8 データ9 データ10 データ11
データ12 データ13 データ14 データ15
DMA 転送
メモリ
51
ビット番号
31 16 15 0
アドレス y+0
アドレス y+4
アドレス y+8
アドレス y+12
ビット7 ビット6
ビット5 ビット4
ビット3 ビット2
ビット1 ビット0
ロード
レジスタ
52
ビット番号
31 16 15 0
ビット7 ビット6

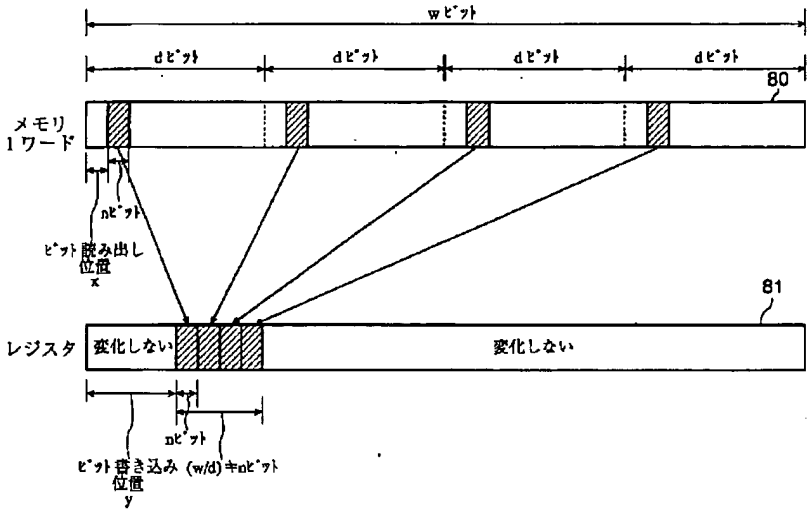
【図７】


メモリ
ビット番号
31 23 15 7
70
アドレス x+0
データ0 データ1 データ2 データ3
ロード命令 (1)
レジスタ
ビット番号
31
74
71
アドレス x+4
データ4 データ5 データ6 データ7
ロード命令 (2)
27
72
アドレス x+8
データ8 データ9 データ10 データ11
ロード命令 (3)
23
73
アドレス x+12
データ12 データ13 データ14 データ15
ロード命令 (4)
19

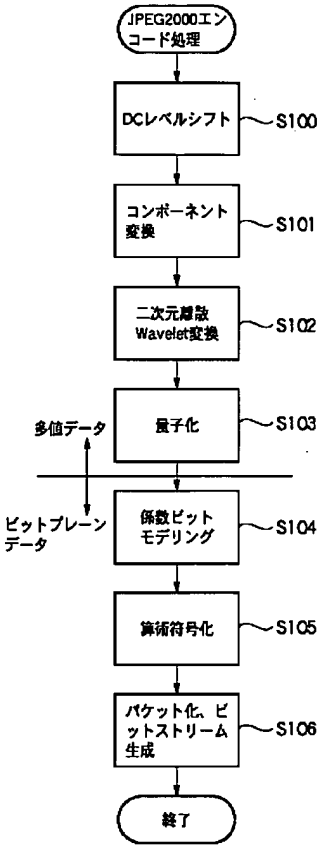
【図９】


BLOAD命令 処理フロー
S90
メモリのアドレスsr+offsetから1ワード読み出し
S91
ループ終了までの処理をw/d回繰り返し
S92
読み出した1ワードのデータからdビットずつ取り出す
S93
取り出したdビット中からd：1のビットセレクタで、ビット位置d－1－#xの1ビットを取り出す
S94
取り出した1ビットをレジスタdrのビット位置31－#yから順に下位に向かって書き込む
S95
ループ終了
命令終了

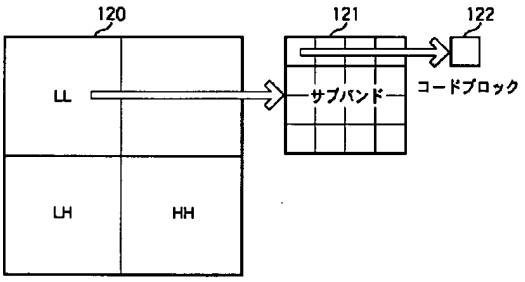
【図８】



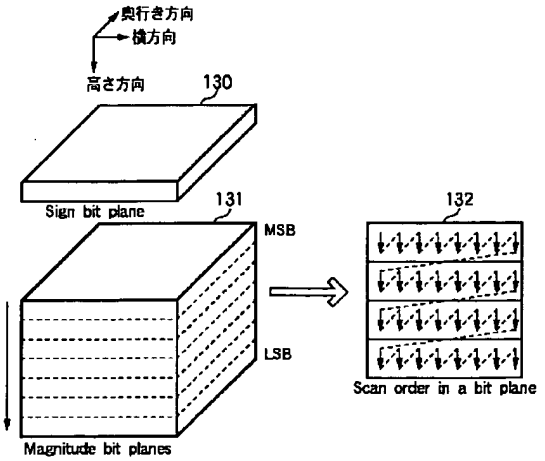
【図１０】



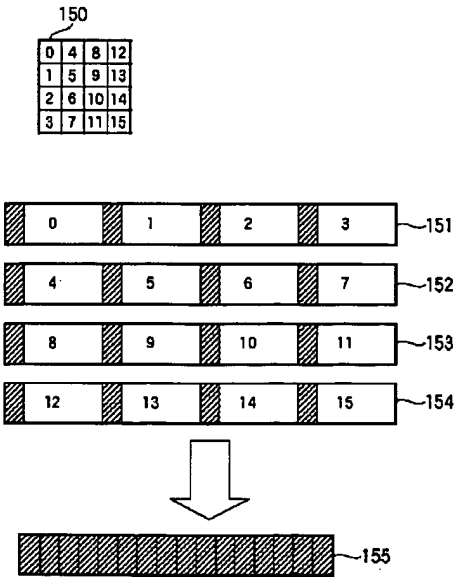
【図１２】



【図１３】



【図１５】



PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-204356

(43)Date of publication of application : 19.07.2002

---

(51)Int.Cl. H04N 1/41
H03M 7/40
H04N 7/30

---

(21)Application number : 2001-297445 (71)Applicant : CANON INC

(22)Date of filing : 27.09.2001 (72)Inventor : OOSA KINYA

---

(30)Priority
Priority number : 2000329421
Priority date : 27.10.2000
Priority country : JP

---

(54) DATA PROCESSING APPARATUS, PROCESSOR AND ITS CONTROL METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To attain speedup and increase in efficiency to transfer a bit plane or bit data especially of multi-valued data.
SOLUTION: In a data processing apparatus, multi-valued data of 8 bits is stored into a memory 50 by 4 pieces per one word, then multi-valued data of 4 by 4=16 pieces is taken

on a single processing unit (a processing block) of a bit plane coding. At memory area 51, the most significant bit (bit 7) of each multi-valued data (in Fig. 5, data 0 - data 15) is gathered in order of each multi-valued data to be stored at one place as shown by a batched part. A bit 6 is also stored in the same way.

LEGAL STATUS [Date of request for examination]
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

1.This document has been translated by computer. So the translation may not reflect the original precisely.

2.**** shows the word which can not be translated.

3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]

[Claim 1] The data processor characterized by carrying out data transfer of the bit plane data constellation which consists of a bit which it is the data processor which performs data transfer between predetermined memory, and two or more multiple-value data constituted by two or more bits exist in the 1st memory, and belongs to the same bit plane from two or more multiple-value data concerned to the 2nd memory per the bit plane data constellation concerned.

[Claim 2] The data processor characterized by generating multiple-value data by choosing the bit data which it is the data processor which performs data transfer between predetermined memory, and two or more bit plane data constellations exist, and belong to the same multiple-value data in each bit plane data constellation in predetermined sequence.

[Claim 3] The data processor according to claim 1 or 2 characterized by containing the data processor according to claim 1 in the coding equipment which performs coding processing including bit plane coding processing.

[Claim 4] Said coding processing is a data processor according to claim 3 characterized by being JPEG2000.

[Claim 5] Said data processor is a data processor given in claim 1 characterized by being a DMA circuit thru/or any 1 term of 4.

[Claim 6] The processor characterized by loading the bit plane data constellation which consists of a bit which it is the processor which loads data to a predetermined register from predetermined memory, and two or more multiple-value data constituted by two or more bits exist in said predetermined memory, and belongs to the same bit plane from two or more multiple-value data concerned to said predetermined register per the bit plane data constellation concerned.

[Claim 7] The control approach of the data processor characterized by carrying out data transfer of the bit plane data constellation which the multiple-value data which are the control approach of a data processor of performing data transfer between predetermined memory, and are constituted by two or more bits

become from the bit which exist in the 1st memory, and belongs to the same bit plane from two or more multiple-value data concerned to the 2nd memory per the bit plane data constellation concerned. [two or more]

[Claim 8] The control approach of the data processor characterized by generating multiple-value data by choosing the bit data which it is the control approach of a data processor of performing data transfer between predetermined memory, and two or more bit plane data constellations exist, and belong to the same multiple-value data in each bit plane data constellation in predetermined sequence.

[Claim 9] The control approach of the processor characterized by loading the bit plane data constellation which the multiple-value data which are the control approach of the processor which loads data to a predetermined register from predetermined memory, and are constituted by two or more bits become from the bit which exist in said predetermined memory, and belongs to the same bit plane from two or more multiple-value data concerned to said predetermined register per the bit plane data constellation concerned. [two or more]

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Field of the Invention] This invention relates to the data processor which performs data transfer between predetermined memory, the processor which loads data to a predetermined register from predetermined memory, and its

control approach.

[0002]

[Description of the Prior Art] As shown in JP,08-149308,A or JP,09-027752,A, the method which carries out bit plane coding (entropy code modulation in a bit plane unit) of the multiplier value after Wavelet conversion is proposed as a method of image coding, and, as for such a method, adoption is planned in the JPEG2000 coding method with which the standardization is advanced in ISO.

[0003] If the flow of processing of this image coding method is generalized and shown, it will become like drawing 2 . drawing 2 -- by being, if the input pixel value 20 carries out conversion coding (21) processing of those with multiple-value data and this to which the value of each pixel is expressed with two or more bits by Wavelet conversion (or other conversion coding methods, such as DCT), the output will serve as a multiplier value (22). It is multiplier value 22 mist beam multiple-value data. Although omitted in this drawing, this multiplier value 22 may be quantized if needed. Next, entropy code modulation (23) of this multiplier value 22 (or quantization value of the result of having quantized this multiplier value) is carried out per bit plane, and coded data (24) is obtained.

[0004] In bit plane coding, in order to have to process a bit plane for the multiple-value data used as an input as a unit, the processing which decomposes an input multiple-value data array into each bit plane is logically needed. Drawing 3 shows the example of the processing which decomposes the array 30 of the 4-bit multiplier value of 4x4 into four bit planes 31, 32, 33, and 34.

[0005] Here, mounting which reads the multiple-value data stored on memory by the coding processor, and performs bit plane (loading to register) coding is considered.

[0006] Suppose the data bus width of face of memory that bit width of face (depth) of w bits and multiple-value data was made into d bits, and d-bit multiple-value data are choked up in order without a clearance on memory in it.

[0007] When a coding processor performs a LDA to this memory, in w bits of data which can be read in one loading, the multiple-value data for a w/d individual are contained, but if its attention is paid only to one bit plane, the bit contained in that bit plane is only a w/d bit, and when considering as the object which uses this bit plane for the following processings, the remaining bits (w-w/d) will turn into a useless bit. Therefore, in order to perform bit plane coding processing to the bit plane of one sheet, it is necessary to repeat loading of a bit

plane repeatedly, to carry out the mask of the unnecessary bit, and to take out a required bit.

[0008] The example at the time of considering as the register width-of-face (1 word) =32 bit of a coding processor, memory data bus width of face of W= 32 bits, and bit width of face of d= 8 bits of multiple-value data in the above-mentioned coding processing at drawing 4 is shown. Every four 8-bit multiple-value data per word are stored in memory 40, and suppose that one batch (processing block) of bit plane coding is made by 4x4=16 piece multiple-value data. Considering the case where the bit plane constituted by the most significant bit (slash section of drawing 4) of multiple-value data is set as the object of bit plane coding processing, first, since 4 bits of data 0 thru/or the most significant bits 31, 23, 15, and 7 of 3 are loaded to a register 41, load processing for 1 word is performed from the address 0 of memory 40.

[0009] Here, a bit required for bit plane coding the loaded data 0 thru/or among 3 is only the above-mentioned 4 bits, and becomes unnecessary [the other bit data]. Therefore, in order to specify a needed bit in this case, bit mask processing must be performed to a register 41, and a required bit must be taken out. Next, further, since the remaining bits under processing block are loaded,

load processing from the address 8 of memory 40 and load processing from the address 12 are performed, and four load processings are needed [since the next data 4 under processing block thru/or the most significant bit of 7 are loaded, load processing is performed from the address 4 of memory 40, and] as the whole processing block.

[0010] Moreover, also when processing of the bit plane of the most significant bit finishes and it progresses to processing of the next bit plane, a bit required of performing four load processings as well as [completely] processing of the bit plane of the most significant bit, and performing bit mask processing to a register 41 is taken out. This processing is repeated to all bit planes (drawing 3 eight sheets).

[0011]

[Problem(s) to be Solved by the Invention] The conventional coding processing by the above explanation requires much a load instruction and bit mask processing of a coding processor, and serves as a failure at the time of attaining increase in efficiency and improvement in the speed of bit plane coding processing. Moreover, even if it mounts processing by the above explanation with software, the result is the same again.

[0012] On the other hand, although above-mentioned JP,08-149308,A and above-mentioned JP,09-027752,A have described the logical algorithm of bit plane coding, the concrete mounting approach is not described.

[0013] To the above trouble, this invention takes an example, and is made, and it aims at performing improvement in the speed of a transfer of the bit plane of bit data, especially multiple-value data, and increase in efficiency.

[0014]

[Means for Solving the Problem] In order to attain the purpose of this invention, the data processor of this invention is equipped with the following configurations. That is, it is the data processor which performs data transfer between predetermined memory, and two or more multiple-value data constituted by two or more bits exist in the 1st memory, and carry out data transfer of the bit plane data constellation which consists of a bit which belongs to the same bit plane from two or more multiple-value data concerned to the 2nd memory per the bit plane data constellation concerned.

[0015] In order to attain the purpose of this invention, the data processor of this invention is equipped with the following configurations. That is, it is the data processor which performs data transfer between predetermined memory, and

two or more bit plane data constellations exist, and multiple-value data are generated by choosing the bit data which belong to the same multiple-value data in each bit plane data constellation in predetermined sequence.

[0016] In order to attain the purpose of this invention, the processor of this invention is equipped with the following configurations. That is, it is the processor which loads data to a predetermined register from predetermined memory, and two or more multiple-value data constituted by two or more bits exist in said predetermined memory, and load the bit plane data constellation which consists of a bit which belongs to the same bit plane from two or more multiple-value data concerned to said predetermined register per the bit plane data constellation concerned.

[0017]

[Embodiment of the Invention] According to an accompanying drawing, this invention is explained to a detail according to a suitable operation gestalt below.

[0018] With the [operation gestalt of ** 1st] book operation gestalt, before the coding processor which is a processor which performs bit plane coding accesses input data (multiple-value data used as the object data of bit plane coding) using a DMA circuit equipped with a multiple-value data-bit plane data conversion

feature, multiple-value data to bit plane data are changed on the memory which stores the input data concerned.

[0019] The processing which the DMA circuit in this operation gestalt performs to drawing 5 is shown. The multiple-value data list on memory 50 is completely the same as that of the multiple-value data list shown in drawing 4 . The DMA circuit in this operation gestalt is transmitted to other fields 51 (it is not necessary to be necessarily the same physical memory as memory 50, and you may be other memory) on memory 50, and performs conversion to bit plane data from multiple-value data format in the case of the transfer.

[0020] In a memory area 51, the most significant bits (bit 7) of each multiple-value data (drawing 5 data 0 thru/or data 15) are collected in order of each multiple-value data (for example, data 0, data 1, data 2, order of **), and as shown by the slash section, it is collectively stored in one place. About a bit 6, it is stored similarly. Thus, by storing each bit in memory, a coding processor is one load processing and can gain all the data (here all data for two bit planes (bits 7 and 6)) of the bit plane to demand. For this reason, the overhead of the processing of a coding processor which became a problem in drawing 4 can be reduced. Moreover, since the data of the bit plane to demand are obtained in 1

lump's form as shown in a register 52, the multiplier value of the near pixel in the same bit plane can be acquired at once, and entropy code modulation using the multiplier value of the near pixel used by JPEG2000 etc. can be mounted efficiently.

[0021] Moreover, although the overhead of a DMA transfer is added even if it introduces the above DMA circuit and the loads of a coding processor decrease in number, processing as the whole system (for example, image coding system using bit plane coding) can be increased the efficiency of and accelerated by making the processing and the DMA transfer which the coding processor is performing to others overlap well. This DMA circuit is carried out in this system as hardware which performs the DMA transfer between memory areas in that case.

[0022] The outline configuration of the coding system which includes this circuit in drawing 1 with the outline configuration of the DMA circuit in this operation gestalt is shown.

[0023] In drawing 1 , the multiple-value data used as input data are stored on main memory 17 in the same format as the example shown by drawing 4 and drawing 5 in the case of bit plane coding, and while the DMA circuit 10 performs

data conversion, the DMA transfer of this multiple-value data is carried out to the local memory 18 for the coding processor 19 as bit plane data from main memory 17. Although DMA data transfer is performed through a bus 16, in this drawing, it is simplified and expressed and this is not necessarily a single bus. (Also when a local memory and a coding processor are in another bus, it can be assumed as main memory.) Moreover, only main memory exists conversely, but also when 17 and 18 are on [both] main memory, they can assume.

How data conversion from main memory 17 to a local memory 18 is made is the same as the example shown in drawing 5 , and the coding processor 19 can load the data (bit plane data) of the bit plane unit stored in the local memory 18, and can perform bit plane coding.

[0024] The internal configuration of the DMA circuit 10 consists of five partial circuits, the source address control circuit 11, the bit-select write-in circuit 12, a word buffer 13, the destination address control circuit 14, and the bus interface circuitry 15, as shown in this drawing.

[0025] The source address control circuit 11 is a circuit which controls the source address of a DMA transfer, and it usually publishes loading of multiple times to main memory 17 in order to fill the data of one batch to the destination to a word

buffer 13. In the example of drawing 1 , since the data for 4 word of the address (x+0) of main memory 17, (x+4), (x+8), and (x+12) are required in order to fill the data stored in the address (y+0) of a local memory 18 to a word buffer 13, four loading is published to main memory 17.

[0026] The bit-select write-in circuit 12 is a circuit which chooses the bit equivalent to a required bit plane from the data for the source data width of face obtained by the source address control circuit 11, takes out, and writes the value of the selected bit in a word buffer 13.

[0027] Concretely, the case where the data stored in the address (y+0) of a local memory 18 are filled to a word buffer 13 is explained to an example about the processing which the bit-select write-in circuit 12 performs. First, the bit plane which the data (data 0 thru/or 3) of the address (x+0) of main memory 17 are read, among those is stored in the address (y+0) of a local memory 18, i.e., the bit 7 and bit 6 of each multiple-value data (data 0 thru/or 3), is taken out by the bit-select write-in circuit 12, and each of the bit 7 of each multiple-value data and a bit 6 is collected and stored in a word buffer 13. Next, the data of the address (x+4) of main memory 17 are read, the bit 7 and bit 6 of each multiple-value data are taken out by the bit-select write-in circuit 12, each of the bit 7 of each

multiple-value data and a bit 6 is brought together in a word buffer 13, and it stores in the bit group loaded from the address (x+0) stored previously adjacently. After processing is similarly repeated to the address (x+8) of main memory, and (x+12) and the processing to the address (x+12) of main memory 17 finishes, the value written in the address (y+0) of a local memory 18 is stored in the word buffer 13.

[0028] A word buffer 13 is a data buffer with the data bus width of face of the destination (local memory 18), and when the bit collected by performing loading from the source (main memory 17) is stored and loading of the multiple times from the source is completed, it holds the data stored in the destination (local memory 18).

[0029] The destination address control circuit 14 is a circuit which controls the address of the data transfer point which a word buffer 13 stores. The source address control circuit 11 publishes four loading to the address (x+0), (x+4), (x+8), and (x+12), and after maintenance of the data stored in a word buffer 13 to the destination (local memory 18) is completed, the destination address control circuit 14 publishes the writing to the address (y+0).

[0030] The bus interface circuitry 15 is a circuit which controls the access timing

to the bus 16 of the DMA circuit 10, and performs actual data transfer on a bus according to the demand from the source address control circuit 11 and the destination address control circuit 14.

[0031] Although drawing 1 has shown the example at the time of using the same parameter as the parameters (memory bus width of face of 32 bits, bit width of face of 8 bits of multiple-value data, etc.) assumed in drawing 3 and drawing 4 By adopting a variable parameter the control approach of the source address control circuit 11, how writing in the method of the bit select of the bit-select write-in circuit 12, and the bit to a local memory 18, etc., and carrying out setting possible It is also possible to constitute the DMA circuit 10 so that it can respond to the data format of various memory bus width of face and multiple-value data, and the batch (magnitude of a processing block) of multiple-value data.

[0032] The DMA circuit in this operation gestalt, and its approach To each multiple-value data stored in main memory by generating bit group data for each [which constitutes the multiple-value data concerned] bit of every, and storing in a local memory for every bit group data of this as the above explanation While being able to gain all the data of the bit plane to demand and being able to simplify bit plane coding processing by one load processing, shortening of the

processing time which this processing takes is attained.

[0033] [the 2nd operation gestalt] -- although the DMA circuit used with the 1st operation gestalt was used on the occasion of bit plane coding processing, the DMA circuit concerned can be used for it also in the case of bit plane decryption processing of hard flow. This operation gestalt explains the decode processing in the decode system which used this DMA circuit.

[0034] Generally bit plane decryption processing will follow and process the data flow shown in drawing 2 to hard flow. That is, a bit plane decryption generates bit plane data for coded data first. Since this decrypted bit plane data is generated per bit plane, it is necessary to return it to a multiple-value data array. after returning to a multiple-value data array, reverse quantization and if required -- inverse transformation coding are performed, and the pixel value of a basis is reproduced. In the flow of a decryption of this single string, in case multiple-value data are generated from the generated bit plane data, said DMA circuit can be used.

[0035] As shown in drawing 6 , the data immediately after a bit plane decryption are a bit plane unit, and are stored in memory 60 as this drawing. Drawing 6 is drawing in which the DMA circuit in this operation gestalt shows the processing

performed from bit plane data in the case of multiple-value data generation.

[0036] Based on the bit plane data for every bit plane stored in memory 60, the bit equivalent to the specific location of each bit plane can be rearranged in memory 61 by collecting by the bit depth of multiple-value data at a multiple-value data array as this drawing.

[0037] The example of this drawing explains conversion actuation of the DMA circuit at the time of a decryption. The data of bits (plane) 7 and 6 exist in the address (y+0) of memory 60, and a DMA circuit loads these bits 7 and 6 to it. In the address (x+0) of the memory 61 which is the destination The inside of the above-mentioned data loaded from the address (y+0) since the multiple-value data for four pieces (data 0 thru/or 3) were collected and stored, The part (a bit 7 the 31 to 28 bit bit number 6 bit numbers 15-12) equivalent to the multiple-value data 0 shown in the slash section thru/or 3 is taken out in the bit-select write-in circuit 12, bits are collected for every multiple-value data, and it writes in a word buffer 13. This actuation is repeatedly performed also to the address (y+4), (y+8), and (y+12). Consequently, it is read into the buffer 13 from the high order bit of each data 0, 1, 2, and 3, and, finally each data 0, 1, 2, and 3 will be restored within a buffer.

[0038] Since the data stored in the address (x+0) of a memory area 61 are stored in the word buffer 13 when the processing to the address (y+12) is completed, the data of the data (data 0 thru/or 3) stored in the word buffer 13 are stored in the address (x+0). Hereafter, the same processing is repeated also to the address (x+4) of a memory area 61, (x+8), and (x+12), and, finally the multiple-value data array of 4x4 is generated.

[0039] The DMA circuit in this operation gestalt and its approach can restore multiple-value data from bit plane data by the above explanation.

[0040] A [operation gestalt of ** 3rd] book operation gestalt explains the processing at the time of loading the bit contained in each bit plane to a register from the memory which stored two or more multiple-value data.

[0041] Moreover, with this operation gestalt, the bit-select write-in circuit 12 carries out in the 1st operation gestalt. From the data for the source data width of face obtained by the source address control circuit 11 The processing which chooses and takes out the bit equivalent to a required bit plane is included in the function of processors (for example, CPU, DSP, etc.) as a bit-select load instruction also including the processing loaded to a register. By the processor concerned The data of a bit plane required for loading and coincidence of

multiple-value data from memory are chosen, and it loads to a register. Of course, it is clear that load processing of the bit plane data to an above-mentioned register may be performed to others in the DMA circuit in the 1st operation gestalt.

[0042] The situation of data processing at the time of using said bit-select load instruction at the time of using the same parameters (memory bus width of face of 32 bits, bit width of face of 8 bits of multiple-value data, etc.) as drawing 3 and the parameter assumed in 4 is shown in drawing 7 . Memory 70, 71, and 72 and the multiple-value data list on 73 are completely the same as the case of the operation gestalt of the 1st and 2. Drawing 7 as well as the operation gestalt of the 1st and 2 shows the bit of the top bit plane in the slash section paying attention to the top bit plane.

[0043] If the top bit plane is specified and the 1st bit-select load instruction is executed, after reading the data for 1 word to memory 70, only the bits of the top bit plane will be chosen and collected and it will write in the bit position pinpointed by the bit numbers 31, 30, 29, and 28 of a register 74. Similarly, the 2nd bit-select load instruction is executed to memory 71 and a register 74. However, the bit of the top bit plane is written in the bit position pinpointed by the

bit numbers 27, 26, 25, and 24 of a register 74 among the bits of the bit plane read from memory 71 by specifying that it is the bit position which shifts the write-in location of the bit of the bit plane to a register 74 with the 1st time shortly, and is pinpointed by the bit number 27. By executing the 3rd time and the 4th bit-select load instruction to the same register similarly hereafter, changing a memory address, finally, a total of the 16-bit data corresponding to the top bit plane will solidify on one register, and they will be arranged.

[0044] Although the count of reading from memory (70, 71, 72, 73) by the case where the usual load instruction is used as a result, and the case where an above-mentioned bit-select load instruction is used is the same Since bit plane coding is performed at the following step, the way which used the bit-select load instruction which can be loaded to a register for every bit plane the data read from memory (70, 71, 72, 73) The utilization ratio of the bit in the register on a processor is high, and there is little register consumption for bit plane data storage, and it ends. Moreover, although the bits of the bit plane to which its attention is paid are scattered in the register after load instruction activation and complicated bit mask processing is needed in the usual load instruction, since the bit of the bit plane to which its attention is paid solidifies and is arranged like

the register 52 of drawing 5 , bit mask processing is also easy for the register after bit-select load instruction use. moreover -- or a bit required only of shift operation can also be taken out as a carry bit, without using bit mask processing.

[0045] The situation of the bit read into a register from memory is shown by executing the bit-select load instruction in this operation gestalt to drawing 8 . In this drawing, d-bit multiple-value data continue and exist each on memory 80, and WORD width of face of a processor is made into w bits.

[0046] A bit-select load instruction reads w bits of data for 1 word from the address position as which memory 80 was specified first. then, every out of the multiple-value data contained in these w bits -- n= 1 bit is chosen, it all comes out, w/dxn=w/dx1=w/d bits are collected, and it writes in to a register 81. At drawing 8 , the slash section shows the selected bit.

[0047] With the register 81, no values other than the part of the register 81 with which the writing of the selected bit is performed change with activation of a bit-select load instruction, but hold the former value.

[0048] Which bit plane is chosen from multiple-value data d bits each on memory 80 specifies with the bit read-out location x (0 <=x<d). Moreover, in a register 81, in which bit position the selected bit is written chooses with the bit write-in

location y (0 <=y<w).

[0049] x and y are BLOAD, when it is given as an operand to a bit-select load instruction, for example, the mnemonic of a bit-select load instruction is set to BLOAD. It can describe on an assembler in a format like dr, [sr+offset] #x, and #y. Here, sr is an address base register which specifies the load address on memory 80, and the contents of sr and the sum of offset serve as a load address.

[0050] dr shows the register of a loading place. # The immediate operand as which x specifies the bit read-out location on memory, and #y are immediate operands which specify the bit write-in location to a register. An example specifies x and y to the last by the immediate operand, and other addressing methods, such as a register operand, may be used for it.

[0051] It is as follows when the assembly code of the example of processing explained using drawing 7 is described using the above-mentioned assembler recording mode. Here, r1 stores the load address on memory 80, and a load result goes into r0 (register).

[0052]

BLOAD r0, [r1] #0, and #0BLOAD r0, [r1+4] #0, and #4BLOAD r0, [r1+8] #0, and #8BLOAD r0, [r1+12] #0, and #12 -- here Although WORD width of face of w bits

and bit width of face of d bits of multiple-value data were given as a regular thing It is possible for these values to also define an instruction as being made to adjustable, and to mount them, and w and d can define and use two or more bit-select load instructions corresponding to the value of w which gave as an operand of a bit-select load instruction, or is different, or d in that case.

[0053] Moreover, although this operation gestalt has shown the case where n= 1 bit of d-bit multiple-value data is chosen for the ejection of one bit plane, also when choosing two or more bits (n> 1) out of each multiple-value data, the easily extensible thing is clear.

[0054] Moreover, the processing flow of a BLOAD instruction is explained using the flow chart of drawing 9 . Drawing 9 is the above-mentioned explanation and is BLOAD. It is explanation of a processing flow at the time of executing the instruction dr, [sr+offset] #x, and #y (n= 1 and assumption).

[0055] First, at step S90, after calculating sr+offset, the data for 1 word are read from memory address sr+offset.

[0056] Next, if it applies to step S95 from step S91, processing which takes out a required bit out of the data for 1 word is performed. Step S91 is an initiation step of the loop formation of bit ejection processing, step S95 is a termination step of

a loop formation, and a loop formation repeats only the count which broke bit width of face (w bits) of 1 word by bit width of face (d bits) of multiple-value data.

[0057] At step S92 of the beginning in a loop formation, d bits equivalent to one multiple-value data are taken out out of 1 word read from memory. Next, at step S93, the bit specified by bit-position #x specified as an operand of a BLOAD instruction is chosen from the d bits using the bit selector of d to 1. Here, temporarily, since bit-position #x are counted from the d-bit high order, they take out 1 bit of the bit position (d-1-x). It writes in at the following step S94, shifting 1 bit taken out at step S93 in low order (the bit position 0 is the least significant) for every loop formation sequentially from the bit position (31-#y) of the register dr of the destination. Namely, by the first loop formation, it writes in the bit position (31-#y) of Register dr, and in the following loop formation, it writes in the bit position (31-#y-1), and writes in hereafter by shifting the bit position written in for every loop formation in the direction of low order.

[0058] After processing of a loop formation finishes the number of fixed times, the writing of the subdevice bit to Register dr is ended, and a BLOAD instruction is ended.

[0059] In addition, although it is the explanation which applies to step S95 from

step S91 in this flow chart, takes out 1 bit at a time and is written in by one loop formation, this is logical explanation, and the mounting top by hardware can take out and write in two or more bits at once by preparing two or more bit selectors of step S93. If w/d bit selectors are prepared and juxtaposition is operated, the processing within a loop formation can be managed at a time.

[0060] [the 4th operation gestalt] -- since it supplements with explanation of all above-mentioned operation gestalten of operation next, the coding processing by JPEG2000 is explained.

[0061] Drawing 10 is drawing having shown the flow of the whole coding processing by JPEG2000. The first step S100 is DC level shift processing, when the pixel value of an input image is expressed with a value without a sign, subtracts a constant from a pixel value and changes it into a value with a sign. When the pixel value of an input image is expressed with a value with a sign, nothing is performed at this step.

[0062] The following step S101 is component transform processing, and changes the value of the color component of three colors by constant matrix multiplication processing of a three dimension. It is processing equivalent to the processing which changes the color expression of a RGB format into the color

expression of a YUV format, and efficient compression is enabled by changing distribution of the value of a color component. Since it is an option, even if it carries out on the occasion of coding processing, it is not necessary to perform step S101.

[0063] Next, at step S102, 2-dimensional discrete wavelet transform (DWT) is performed to a pixel value. First, if DWT of a single dimension is explained, two kinds of filters, 5-3 Reversible filter and 9-7 Ireversible filter, will be defined by JPEG2000, and it will change by it using either. Here, since it is easy, 5-3 Reversible filter is explained. the input value to a single dimension DWT -- X (n) -- (-- n= -- 0, 1, --, N-1) -- carrying out -- an output value -- Y (n) -- (-- n= -- when 0, 1, --, N-1), processing of 5-3 Reversible filter is expressed with two steps of following formulas.

[0064]

2Y(2n) =X(2n)- [Y(2n+1) =X(2n+1)- (X(2n)+X (2n+2)) /] (Y(2n-1)+Y(2n+1)+2)/4 -- here When the maximum integer which does not exceed x shall be expressed and the range of n of an input value X (n) exceeds range 0- (N-1) from the first, after [x] extends the range where X (n) is defined by the approach defined by specification, it is used. Y (2n) is equivalent to the multiplier of a low frequency

(L) component, and Y (2n+1) is equivalent to the multiplier of a RF (H) component.

[0065] 9-7 Although in Ireversible filter a formula increases to six steps, and the value of the constant multiplier in a formula also changes and it becomes complicated, the fundamental count approach in each phase is the same. Even if it uses which filter, the output value of N individual is generated from the input value of N individual, and, as for N/2 piece which is the eventh among output values, a low frequency (L) component and the oddth N/2 are RF (H) components.

[0066] Next, two dimensions are explained using drawing 11 about processing of DWT. In drawing 11 , to the 2-dimensional input pixel (group) 110, first, vertical single dimension DWT processing is performed, and it is collecting the output by the side of the low frequency (L) of a single dimension DWT, and the outputs by the side of a RF (H), and becomes the multiplier value 111 after perpendicular DWT for every train. Next, for every line of the multiplier value 111, horizontal single dimension DWT processing is performed and the output staff numeric value 112 after 2-dimensional DWT is similarly acquired by collecting the output coefficients by the side of L and H. In 112, the part in which LH subband, and a

horizontal and a perpendicular brought together the component by the side of H for the part in which HL subband and the perpendicular direction brought together the part in which LL subband and the perpendicular direction brought together the part in which the horizontal and the perpendicular brought together the component by the side of L, and L component and the horizontal direction brought H component together, and H component and the horizontal direction brought L component together is called HH subband. About LL subband, it is repeating and giving 2 more-dimensional DWT, and usually divides into a finer subband.

[0067] At step S103, quantization processing of the multiplier of each divided subband is performed. When an input staff numeric value is set to x and a quantization step is set to delta, the multiplier value q after quantization is q=sign(x) x [abs(x) / delta].

It becomes. Here, sign (x) is the sign of x, 1 is returned at the time of -1 and x>=0 at the time of x< 0, and abs (x) returns the absolute value of x. [x] is the maximum integer which does not exceed x. Quantization step delta is a fixed value for every subband.

[0068] Furthermore, although a pixel value and a multiplier value are expressed

with a two-complement-form type on account of count in many cases, if step S103 is missing from the following step S104 from step S103, since it is necessary to divide the sign and absolute value of a multiplier separately and it needs to process them, it is changed into the sign and absolute value format expressed in the remaining bits which show 1 bit which shows the sign of a multiplier for the transcription of a multiplier value, and an absolute value.

[0069] Although multiplier bit modeling processing is performed at the following step S104, to processing of steps S102 and S103 being processing to a subband, processing of step S104 divides a subband further, and processes in the unit of a code block.

[0070] Drawing 12 is the image multiplier data 120 after 2-dimensional DWT, the subband 121, and drawing having shown the relation of the code block 122, one subband 121 is taken out and quantized among the multiplier data after 2-dimensional DWT (step S102) (step S103), the subband 121 is divided still like a dotted line, and every one of them is processed at step S104 as code block 122.

[0071] Moreover, in drawing 10 , to step S103, an above-mentioned operation gestalt is applied in the part of the data input of step S104 in order to

disassemble multiple-value data into a bit plane at step S104 to treating a pixel value and a multiplier value as data of a multiple value and to process as bit plane data.

[0072] The processing sequence of the data input of step S104 is explained using drawing 13 . The drawing 13 chart on the left is drawing having shown the multiple-value data (a sign and absolute value format) of a code block, and shows the breadth on the space of a code block of the longitudinal direction and the depth direction in drawing, and the height direction in drawing shows the bit depth of multiple-value data. 130 is a bit plane which consists only of a sign bit of a multiplier, and 131 is the bit which shows the absolute value of a multiplier, and is divided into the bit plane for every bit depth by the dotted line. If the absolute value bit shall be located in a line toward the bottom from drawing Nakagami (to LSB) (from MSB), the absolute value bit plane 131 will be processed in order toward LSB from MSB. The sign bit plane 130 is referred to for processing, as processing of the absolute value bit plane 131 progresses, and a sign bit is also processed according to conditions.

[0073] Drawing on the right-hand side of drawing 13 shows the processing sequence of the bit in the bit plane, when one bit plane 132 is taken out from the

absolute value bit plane 131. A bit plane 132 is divided for every 4 bits long, and is processed according to the sense of the arrow head of right-hand side drawing. Whenever the small arrow head of a lengthwise direction shows 4 bits long and 4 bits long processing finishes once, 4 bits of right-hand are processed from a top to the bottom. If processing arrives at a right end, 4 bits long of the following line are processed, and this scan will be hereafter performed 3 times in one bit plane until the bit plane data corresponding to all code blocks are processed. The reason three scans are performed is that there are a bit used as a processing object and a bit not becoming, and it scans 3 times according to conditions with each three scan, and is because processing of all bits is completed. The selection condition of the object bit in each scan is specified on specification.

[0074] At step S104, count of the context used by the next algebraic-sign-ized processing (step S105) and the bit value encoded is performed, scanning a multiplier bit in the sequence shown in drawing 13 . The bit value of the multiplier bit 140 which serves as a candidate for coding as shown in drawing 14 , and the that about eight perimeter multiplier bit 141 (slash section), and the condition of being calculated along with it are used for this count. In drawing 14 , if it is the list

whose four lines of a center are 4 bits long under present processing, 142 belongs to 4 bits long in front of one, 143 belongs to one 4 bits long as follows, but since about eight perimeter of an object pixel is needed, the value of 142 and the multiplier bit of 143 and status information are required also for processing of 4 bits long under present processing. After referring to the bit information on these many, a context and the bit value encoded are calculated and is passed to the following algebraic-sign-ized step S105.

[0075] In drawing 10 , by algebraic-sign-ized processing of step S105, algebraic-sign-ization of the context base is performed by considering the context calculated at step S104, and the bit value encoded as an input, and the amount of data of multiplier data is compressed.

[0076] At the following step S106, the data compressed at step S105 are packet-ized in a suitable unit, a required header segment is added, and an output bit stream is generated.

[0077] Coding processing of JPEG2000 is ended by the above processing. In addition, decryption processing follows the procedure of drawing 10 conversely, and can be realized by performing inverse transformation of each step. It is multiplier bit modeling processing of step S104 that an above-mentioned

operation gestalt is applied in the procedure of the coding processing of JPEG2000 shown by drawing 10 . Since the data of a processing object change from this step from multiple-value data to bit plane data, processing of step S104 can be efficiently performed by performing transform processing by the above-mentioned operation gestalt.

[0078] The example at the time of applying an above-mentioned operation gestalt to drawing 15 at step 104 is shown. Suppose that the processing object in step S104 is the multiple-value multiplier data 150 of 4x4 in drawing 15 . Although size of multiplier data was set to 4x4 in order to simplify explanation, it becomes actual more big size. As drawing 13 explained, the multiplier data 150 are numerical order in drawing (0, 1, 2, --, 15), and are processed toward a low order bit plane from a high order bit plane. Here, the condition in the time of this multiplier data being on memory (for example, main memory 17) is shown in 151, 152, 153, and 154. 151,152,153,154 is a 1-word storage region on memory, respectively, and multipliers 0-3, multipliers 4-7, multipliers 8-11, multipliers 12-15, and with a multipliers [every] of four data are contained in each. Supposing the best bit of each multiple-value multiplier data is a sign bit, it turns out among drawing that the slash section is equivalent to a sign bit, and

distributes and exists in each memory WORD. If processing which collects the bits which are in the location of a sign bit from each WORD about this according to the 1st above-mentioned operation gestalt or the 3rd operation gestalt is performed, the data 155 with which only sign bits gathered on the local memory or the register can be obtained as a result.

[0079] Once it can collect sign bits in the form of data 155, henceforth, it can obtain easily only by calculating the AND (AND) of a bit mask and data 155, and also when the value of the sign bit of a multiplier 0 - 15 throats is required, as shown in drawing 14 , the efficiency of the processing which acquires the value of the data of the near pixel of the bit for coding can be increased. Since especially the sign bit plane 130 is needed all the time during processing of the absolute value bit plane 131 as drawing of the drawing 13 left showed, the bit of the arbitration of the data 155 can be efficiently referred to by storing like the data 155 of drawing 15 .

[0080]

[Effect of the Invention] According to this invention, improvement in the speed of a transfer of the bit plane of bit data, especially multiple-value data and increase in efficiency can be performed by the above explanation.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] With the outline configuration of the DMA circuit in the 1st operation gestalt of this invention, the outline configuration of a coding system including

this circuit is shown.

[Drawing 2] It is drawing showing the flow of processing of the bit plane coding method of the generalized image.

[Drawing 3] It is drawing showing the example of the processing decomposed into four bit planes 31, 32, 33, and 34 of the array 30 of the 4-bit multiplier value of 4x4.

[Drawing 4] In the conventional coding processing, it is drawing showing the example at the time of considering as the register width-of-face (1 word) =32 bit of a coding processor, memory data bus width of face of W= 32 bits, and bit width of face of d= 8 bits of multiple-value data.

[Drawing 5] It is drawing showing the processing which the DMA circuit in the 1st operation gestalt of this invention performs.

[Drawing 6] The DMA circuit in the 2nd operation gestalt of this invention is drawing showing the processing performed from bit plane data in the case of multiple-value data generation.

[Drawing 7] It is drawing showing the situation of data processing at the time of using a bit-select load instruction.

[Drawing 8] It is drawing showing the situation of the bit read into a register from

memory by executing a bit-select load instruction.

[Drawing 9] It is the processing flow chart of a BLOAD instruction.

[Drawing 10] It is the flow chart of the coding processing in JPEG2000.

[Drawing 11] It is drawing explaining processing of two-dimensional DWT.

[Drawing 12] They are the image multiplier data 210 after two-dimensional DWT, the subband 121, and drawing showing the relation of the code block 122.

[Drawing 13] It is drawing explained about the processing sequence of the data input of step S104.

[Drawing 14] It is drawing showing the multiplier bit 140 used as the candidate for coding, and the about eight perimeter multiplier bit 141 (slash section).

[Drawing 15] It is drawing showing the example at the time of applying the operation gestalt of this invention to step S104.